

# The Perceptron with Dynamic Margin

Constantinos Panagiotakopoulos and Petroula Tsampouka

Physics Division, School of Technology  
Aristotle University of Thessaloniki, Greece  
costapan@eng.auth.gr, petroula@gen.auth.gr

**Abstract.** The classical perceptron rule provides a varying upper bound on the maximum margin, namely the length of the current weight vector divided by the total number of updates up to that time. Requiring that the perceptron updates its internal state whenever the normalized margin of a pattern is found not to exceed a certain fraction of this dynamic upper bound we construct a new approximate maximum margin classifier called the perceptron with dynamic margin (PDM). We demonstrate that PDM converges in a finite number of steps and derive an upper bound on them. We also compare experimentally PDM with other perceptron-like algorithms and support vector machines on hard margin tasks involving linear kernels which are equivalent to 2-norm soft margin.

**Keywords:** Online learning, classification, maximum margin.

## 1 Introduction

It is a common belief that learning machines able to produce solution hyperplanes with large margins exhibit greater generalization ability [21] and this justifies the enormous interest in Support Vector Machines (SVMs) [21, 2]. Typically, SVMs obtain large margin solutions by solving a constrained quadratic optimization problem using dual variables. In their native form, however, efficient implementation is hindered by the quadratic dependence of their memory requirements in the number of training examples a fact which renders prohibitive the processing of large datasets. To overcome this problem decomposition methods [15, 6] were developed that apply optimization only to a subset of the training set. Although such methods led to improved convergence rates, in practice their superlinear dependence on the number of examples, which can be even cubic, can still lead to excessive runtimes when large datasets are processed. Recently, the so-called linear SVMs [7, 8, 13] made their appearance. They take advantage of linear kernels in order to allow parts of them to be written in primal notation and were shown to outperform decomposition SVMs when dealing with massive datasets.

The above considerations motivated research in alternative large margin classifiers naturally formulated in primal space long before the advent of linear SVMs. Such algorithms are mostly based on the perceptron [16, 12], the simplest online learning algorithm for binary linear classification. Like the perceptron, they focus on the primal problem by updating a weight vector which represents

at each step the current state of the algorithm whenever a data point presented to it satisfies a specific condition. It is the ability of such algorithms to process one example at a time<sup>1</sup> that allows them to spare time and memory resources and consequently makes them able to handle large datasets. The first algorithm of that kind is the perceptron with margin [3] which is much older than SVMs. It is an immediate extension of the perceptron which provably achieves solutions with only up to 1/2 of the maximum margin [10]. Subsequently, various algorithms succeeded in approximately attaining maximum margin by employing modified perceptron-like update rules. Such algorithms include ROMMA [11], ALMA [5], CRAMMA [19] and MICRA [20]. Very recently, the same goal was accomplished by a generalized perceptron with margin, the margitron [14].

The most straightforward way of obtaining large margin solutions through a perceptron is by requiring that the weight vector be updated every time the example presented to the algorithm has (normalized) margin which does not exceed a predefined value [17, 18, 1]. The obvious problem with this idea, however, is that the algorithm with such a fixed margin condition will definitely not converge unless the target value of the margin is smaller than the unknown maximum margin. In an earlier work [14] we noticed that the upper bound  $\|\mathbf{a}_t\|/t$  on the maximum margin, with  $\|\mathbf{a}_t\|$  being the length of the weight vector and  $t$  the number of updates, that comes as an immediate consequence of the perceptron update rule is very accurate and tends to improve as the algorithm achieves larger margins. In the present work we replace the fixed target margin value with a fraction  $1 - \epsilon$  of this varying upper bound on the maximum margin. The hope is that as the algorithm keeps updating its state the upper bound will keep approaching the maximum margin and convergence to a solution with the desired accuracy  $\epsilon$  will eventually occur. Thus, the resulting algorithm may be regarded as a realizable implementation of the perceptron with fixed margin condition.

The rest of this paper is organized as follows. Section 2 contains some preliminaries and a motivation of the algorithm based on a qualitative analysis. In Sect. 3 we give a formal theoretical analysis. Section 4 is devoted to implementational issues. Section 5 contains our experimental results while Sect. 6 our conclusions.

## 2 Motivation of the Algorithm

Let us consider a linearly separable training set  $\{(\mathbf{x}_k, l_k)\}_{k=1}^m$ , with vectors  $\mathbf{x}_k \in \mathbb{R}^d$  and labels  $l_k \in \{+1, -1\}$ . This training set may either be the original dataset or the result of a mapping into a feature space of higher dimensionality [21, 2]. Actually, there is a very well-known construction [4] making linear separability always possible, which amounts to the adoption of the 2-norm soft margin. By placing  $\mathbf{x}_k$  in the same position at a distance  $\rho$  in an additional dimension, i.e. by extending  $\mathbf{x}_k$  to  $[\mathbf{x}_k, \rho]$ , we construct an embedding of our data into the so-called augmented space [3]. This way, we construct hyperplanes possessing bias

---

<sup>1</sup> The conversion of online algorithms to the batch setting is done by cycling repeatedly through the dataset and using the last hypothesis for prediction.

in the non-augmented feature space. Following the augmentation, a reflection with respect to the origin of the negatively labeled patterns is performed by multiplying every pattern with its label. This allows for a uniform treatment of both categories of patterns. Also,  $R \equiv \max_k \|\mathbf{y}_k\|$  with  $\mathbf{y}_k \equiv [l_k \mathbf{x}_k, l_k \rho]$  the  $k^{\text{th}}$  augmented and reflected pattern. Obviously,  $R \geq \rho$ .

The relation characterizing optimally correct classification of the training patterns  $\mathbf{y}_k$  by a weight vector  $\mathbf{u}$  of unit norm in the augmented space is

$$\mathbf{u} \cdot \mathbf{y}_k \geq \gamma_d \equiv \max_{\mathbf{u}' : \|\mathbf{u}'\|=1} \min_i \{\mathbf{u}' \cdot \mathbf{y}_i\} \quad \forall k. \quad (1)$$

We shall refer to  $\gamma_d$  as the maximum directional margin. It coincides with the maximum margin in the augmented space with respect to hyperplanes passing through the origin. For the maximum directional margin  $\gamma_d$  and the maximum geometric margin  $\gamma$  in the non-augmented feature space, it holds that  $1 \leq \gamma/\gamma_d \leq R/\rho$ . As  $\rho \rightarrow \infty$ ,  $R/\rho \rightarrow 1$  and, consequently,  $\gamma_d \rightarrow \gamma$  [17, 18].

We consider algorithms in which the augmented weight vector  $\mathbf{a}_t$  is initially set to zero, i.e.  $\mathbf{a}_0 = \mathbf{0}$ , and is updated according to the classical perceptron rule

$$\mathbf{a}_{t+1} = \mathbf{a}_t + \mathbf{y}_k \quad (2)$$

each time an appropriate misclassification condition is satisfied by a training pattern  $\mathbf{y}_k$ . Taking the inner product of (2) with the optimal direction  $\mathbf{u}$  and using (1) we get

$$\mathbf{u} \cdot \mathbf{a}_{t+1} - \mathbf{u} \cdot \mathbf{a}_t = \mathbf{u} \cdot \mathbf{y}_k \geq \gamma_d$$

a repeated application of which gives [12]

$$\|\mathbf{a}_t\| \geq \mathbf{u} \cdot \mathbf{a}_t \geq \gamma_d t. \quad (3)$$

From (3) we readily obtain

$$\gamma_d \leq \frac{\|\mathbf{a}_t\|}{t} \quad (4)$$

provided  $t > 0$ . Notice that the above upper bound on the maximum directional margin  $\gamma_d$  is an immediate consequence of the classical perceptron rule and holds independent of the misclassification condition.

It would be very desirable that  $\|\mathbf{a}_t\|/t$  approaches  $\gamma_d$  with  $t$  increasing since this would provide an after-run estimate of the accuracy achieved by an algorithm employing the classical perceptron update. More specifically, with  $\gamma'_d$  being the directional margin achieved upon convergence of the algorithm in  $t_c$  updates, it holds that

$$\frac{\gamma_d - \gamma'_d}{\gamma_d} \leq 1 - \frac{\gamma'_d t_c}{\|\mathbf{a}_{t_c}\|}. \quad (5)$$

In order to understand the mechanism by which  $\|\mathbf{a}_t\|/t$  evolves we consider the difference between two consecutive values of  $\|\mathbf{a}_t\|^2/t^2$  which may be shown to be given by the relation

$$\frac{\|\mathbf{a}_t\|^2}{t^2} - \frac{\|\mathbf{a}_{t+1}\|^2}{(t+1)^2} = \frac{1}{t(t+1)} \left\{ \left( \frac{\|\mathbf{a}_t\|^2}{t} - \mathbf{a}_t \cdot \mathbf{y}_k \right) + \left( \frac{\|\mathbf{a}_{t+1}\|^2}{t+1} - \mathbf{a}_{t+1} \cdot \mathbf{y}_k \right) \right\}. \quad (6)$$

Let us assume that satisfaction of the misclassification condition by a pattern  $\mathbf{y}_k$  has as a consequence that  $\|\mathbf{a}_t\|^2/t > \mathbf{a}_t \cdot \mathbf{y}_k$  (i.e., the normalized margin  $\mathbf{u}_t \cdot \mathbf{y}_k$  of  $\mathbf{y}_k$  (with  $\mathbf{u}_t \equiv \mathbf{a}_t/\|\mathbf{a}_t\|$ ) is smaller than the upper bound (4) on  $\gamma_d$ ). Let us further assume that after the update has taken place  $\mathbf{y}_k$  still satisfies the misclassification condition and therefore  $\|\mathbf{a}_{t+1}\|^2/(t+1) > \mathbf{a}_{t+1} \cdot \mathbf{y}_k$ . Then, the r.h.s. of (6) is positive and  $\|\mathbf{a}_t\|/t$  decreases as a result of the update. In the event, instead, that the update leads to violation of the misclassification condition,  $\|\mathbf{a}_{t+1}\|^2/(t+1)$  is not necessarily larger than  $\mathbf{a}_{t+1} \cdot \mathbf{y}_k$  and  $\|\mathbf{a}_t\|/t$  may not decrease as a result of the update. We expect that statistically, at least in the early stages of the algorithm, most updates do not lead to correctly classified patterns (i.e., patterns which violate the misclassification condition) and as a consequence  $\|\mathbf{a}_t\|/t$  will have the tendency to decrease. Obviously, the rate at which this will take place depends on the size of the difference  $\|\mathbf{a}_t\|^2/t - \mathbf{a}_t \cdot \mathbf{y}_k$  which, in turn, depends on the misclassification condition.

If we are interested in obtaining solutions possessing margin the most natural choice of misclassification condition is the fixed (normalized) margin condition

$$\mathbf{a}_t \cdot \mathbf{y}_k \leq (1 - \epsilon)\gamma_d \|\mathbf{a}_t\| \quad (7)$$

with the accuracy parameter  $\epsilon$  satisfying  $0 < \epsilon \leq 1$ . This is an example of a misclassification condition which if it is satisfied ensures that  $\|\mathbf{a}_t\|^2/t > \mathbf{a}_t \cdot \mathbf{y}_k$ . Moreover, by making use of (4) and (7) it may easily be shown that  $\|\mathbf{a}_{t+1}\|^2/(t+1) \geq \mathbf{a}_{t+1} \cdot \mathbf{y}_k$  for  $t \geq \epsilon^{-1}R^2/\gamma_d^2$ . Thus, after at most  $\epsilon^{-1}R^2/\gamma_d^2$  updates  $\|\mathbf{a}_t\|/t$  decreases monotonically. The perceptron algorithm with fixed margin condition (PFM) is known to converge in a finite number of updates to an  $\epsilon$ -accurate approximation of the maximum directional margin hyperplane [17, 18, 1]. Although it appears that PFM demands exact knowledge of the value of  $\gamma_d$ , we notice that only the value of  $\beta \equiv (1 - \epsilon)\gamma_d$ , which is the quantity entering (7), needs to be set and not the values of  $\epsilon$  and  $\gamma_d$  separately. That is why the after-run estimate (5) is useful in connection with the algorithm in question. Nevertheless, in order to make sure that  $\beta < \gamma_d$  a priori knowledge of a fairly good lower bound on  $\gamma_d$  is required and this is an obvious defect of PFM.

The above difficulty associated with the fixed margin condition may be remedied if the unknown  $\gamma_d$  is replaced for  $t > 0$  with its varying upper bound  $\|\mathbf{a}_t\|/t$

$$\mathbf{a}_t \cdot \mathbf{y}_k \leq (1 - \epsilon) \frac{\|\mathbf{a}_t\|^2}{t} . \quad (8)$$

Condition (8) ensures that  $\|\mathbf{a}_t\|^2/t - \mathbf{a}_t \cdot \mathbf{y}_k \geq \epsilon\|\mathbf{a}_t\|^2/t > 0$ . Moreover, as in the case of the fixed margin condition,  $\|\mathbf{a}_{t+1}\|^2/(t+1) - \mathbf{a}_{t+1} \cdot \mathbf{y}_k \geq 0$  for  $t \geq \epsilon^{-1}R^2/\gamma_d^2$ . As a result, after at most  $\epsilon^{-1}R^2/\gamma_d^2$  updates the r.h.s. of (6) is bounded from below by  $\epsilon\|\mathbf{a}_t\|^2/t^2(t+1) \geq \epsilon\gamma_d^2/(t+1)$  and  $\|\mathbf{a}_t\|/t$  decreases monotonically and sufficiently fast. Thus, we expect that  $\|\mathbf{a}_t\|/t$  will eventually approach  $\gamma_d$  close enough, thereby allowing for convergence of the algorithm to an  $\epsilon$ -accurate approximation of the maximum directional margin hyperplane. It is also apparent that the decrease of  $\|\mathbf{a}_t\|/t$  will be faster for larger values of  $\epsilon$ .

---

The Perceptron with Dynamic Margin

---

**Input:** A linearly separable augmented dataset  $S = (\mathbf{y}_1, \dots, \mathbf{y}_k, \dots, \mathbf{y}_m)$  with reflection assumed  
**Fix:**  $\epsilon$   
**Define:**  $q_k = \|\mathbf{y}_k\|^2$ ,  $\bar{\epsilon} = 1 - \epsilon$   
**Initialize:**  $t = 0$ ,  $\mathbf{a}_0 = \mathbf{0}$ ,  $\ell_0 = 0$ ,  $\theta_0 = 0$   
**repeat**  
    **for**  $k = 1$  **to**  $m$  **do**  
         $p_{tk} = \mathbf{a}_t \cdot \mathbf{y}_k$   
        **if**  $p_{tk} \leq \theta_t$  **then**  
             $\mathbf{a}_{t+1} = \mathbf{a}_t + \mathbf{y}_k$   
             $\ell_{t+1} = \ell_t + 2p_{tk} + q_k$   
             $t \leftarrow t + 1$   
             $\theta_t = \bar{\epsilon} \ell_t / t$   
**until** no update made within the **for** loop

---

The perceptron algorithm employing the misclassification condition (8) (with its threshold set to 0 for  $t = 0$ ), which may be regarded as originating from (7) with  $\gamma_d$  replaced for  $t > 0$  by its dynamic upper bound  $\|\mathbf{a}_t\|/t$ , will be named the perceptron with dynamic margin (PDM).

### 3 Theoretical Analysis

From the discussion that led to the formulation of PDM it is apparent that if the algorithm converges it will achieve by construction

a solution possessing directional margin at least as large as  $(1 - \epsilon)\gamma_d$ . (We remind the reader that convergence assumes violation of the misclassification condition (8) by all patterns. In addition, (4) holds.) The same obviously applies to PFM. Thus, for both algorithms it only remains to be demonstrated that they converge in a finite number of steps. This has already been shown for PFM [17, 18, 1] but no general  $\epsilon$ -dependent bound in closed form has been derived. Our purpose in this section is to demonstrate convergence of PDM and provide explicit bounds for both algorithms.

Before we proceed with our analysis we will need the following result.

**Lemma 1.** *Let the variable  $t \geq e^{-C}$  satisfy the inequality*

$$t < \delta(1 + C + \ln t) , \quad (9)$$

where  $\delta, C$  are constants and  $\delta > e^{-C}$ . Then

$$t \leq t_0 \equiv (1 + e^{-1})\delta(C + \ln((1 + e)\delta)) . \quad (10)$$

*Proof.* If  $t \geq e^{-C}$  then  $(1 + C + \ln t) \geq 1$  and inequality (9) is equivalent to  $f(t) = t/(1 + C + \ln t) - \delta < 0$ . For the function  $f(t)$  defined in the interval  $[e^{-C}, +\infty)$  it holds that  $f(e^{-C}) < 0$  and  $df/dt = (C + \ln t)/(1 + C + \ln t)^2 > 0$  for  $t > e^{-C}$ . Stated differently,  $f(t)$  starts from negative values at  $t = e^{-C}$  and increases monotonically. Therefore, if  $f(t_0) \geq 0$  then  $t_0$  is an upper bound of all  $t$  for which  $f(t) < 0$ . Indeed, it is not difficult to verify that  $t_0 > \delta > e^{-C}$  and

$$f(t_0) = \delta \left( (1 + e^{-1}) \left( 1 + \frac{\ln \ln(e^C(1 + e)\delta)}{\ln(e^C(1 + e)\delta)} \right)^{-1} - 1 \right) \geq 0$$

given that  $\ln \ln x / \ln x \leq e^{-1}$ . □

Now we are ready to derive an upper bound on the number of steps of PFM.

**Theorem 1.** *The number  $t$  of updates of the perceptron algorithm with fixed margin condition satisfies the bound*

$$t \leq \frac{(1+e^{-1})}{2\epsilon} \frac{R^2}{\gamma_d^2} \left\{ 4 \frac{\gamma_d}{R} \left( 1 - \frac{\gamma_d}{R} (1-\epsilon) \right) + \ln \left( \frac{(1+e)}{\epsilon} \frac{R}{\gamma_d} \left( 1 - \frac{\gamma_d}{R} (1-\epsilon) \right) \right) \right\} .$$

*Proof.* From (2) and (7) we get

$$\|\mathbf{a}_{t+1}\|^2 = \|\mathbf{a}_t\|^2 + \|\mathbf{y}_k\|^2 + 2\mathbf{a}_t \cdot \mathbf{y}_k \leq \|\mathbf{a}_t\|^2 \left( 1 + \frac{R^2}{\|\mathbf{a}_t\|^2} + \frac{2(1-\epsilon)\gamma_d}{\|\mathbf{a}_t\|} \right) .$$

Then, taking the square root and using the inequality  $\sqrt{1+x} \leq 1+x/2$  we have

$$\|\mathbf{a}_{t+1}\| \leq \|\mathbf{a}_t\| \left( 1 + \frac{R^2}{\|\mathbf{a}_t\|^2} + \frac{2(1-\epsilon)\gamma_d}{\|\mathbf{a}_t\|} \right)^{\frac{1}{2}} \leq \|\mathbf{a}_t\| \left( 1 + \frac{R^2}{2\|\mathbf{a}_t\|^2} + \frac{(1-\epsilon)\gamma_d}{\|\mathbf{a}_t\|} \right) .$$

Now, by making use of  $\|\mathbf{a}_t\| \geq \gamma_d t$ , we observe that

$$\|\mathbf{a}_{t+1}\| - \|\mathbf{a}_t\| \leq \frac{R^2}{2\|\mathbf{a}_t\|} + (1-\epsilon)\gamma_d \leq \frac{R^2}{2\gamma_d} \frac{1}{t} + (1-\epsilon)\gamma_d .$$

A repeated application of the above inequality  $t - N$  times ( $t > N \geq 1$ ) gives

$$\begin{aligned} \|\mathbf{a}_t\| - \|\mathbf{a}_N\| &\leq \frac{R^2}{2\gamma_d} \sum_{k=N}^{t-1} k^{-1} + (1-\epsilon)\gamma_d(t-N) \\ &< \frac{R^2}{2\gamma_d} \left( \frac{1}{N} + \int_N^t k^{-1} dk \right) + (1-\epsilon)\gamma_d(t-N) \end{aligned}$$

from where using the obvious bound  $\|\mathbf{a}_N\| \leq RN$  we get an upper bound on  $\|\mathbf{a}_t\|$

$$\|\mathbf{a}_t\| < \frac{R^2}{2\gamma_d} \left( \frac{1}{N} + \ln \frac{t}{N} \right) + (1-\epsilon)\gamma_d(t-N) + RN .$$

Combining the above upper bound on  $\|\mathbf{a}_t\|$ , which holds not only for  $t > N$  but also for  $t = N$ , with the lower bound from (3) we obtain

$$t < \frac{1}{2\epsilon} \frac{R^2}{\gamma_d^2} \left\{ \frac{1}{N} - \ln N + 2 \frac{\gamma_d}{R} \left( 1 - \frac{\gamma_d}{R} (1-\epsilon) \right) N + \ln t \right\} .$$

Setting

$$\delta = \frac{1}{2\epsilon} \frac{R^2}{\gamma_d^2} , \quad \alpha = 2 \frac{\gamma_d}{R} \left( 1 - \frac{\gamma_d}{R} (1-\epsilon) \right)$$

and choosing  $N = 1 + \lceil \alpha^{-1} \rceil$ , with  $\lceil x \rceil$  being the integer part of  $x \geq 0$ , we finally get

$$t < \delta(1 + 2\alpha + \ln \alpha + \ln t) . \quad (11)$$

Notice that in deriving (11) we made use of the fact that  $\alpha N + N^{-1} - \ln N < 1 + 2\alpha + \ln \alpha$ . Inequality (11) has the form (9) with  $C = 2\alpha + \ln \alpha$ . Obviously,  $e^{-C} < \alpha^{-1} < N \leq t$  and  $e^{-C} < \alpha^{-1} \leq \delta$ . Thus, the conditions of Lemma 1 are satisfied and the required bound, which is of the form (10), follows from (11).  $\square$

Finally, we arrive at our main result which is the proof of convergence of PDM in a finite number of steps and the derivation of the relevant upper bound.

**Theorem 2.** *The number  $t$  of updates of the perceptron algorithm with dynamic margin satisfies the bound*

$$t \leq \begin{cases} t_0 \left(1 - \frac{1}{1-2\epsilon} \frac{R^2}{\gamma_d^2} t_0^{-1}\right)^{\frac{1}{2\epsilon}}, & t_0 \equiv \lceil \epsilon^{-1} \rceil \left(\frac{R}{\gamma_d}\right)^{\frac{1}{\epsilon}} \left(1 + \frac{\lceil \epsilon^{-1} \rceil^{-1}}{1-2\epsilon}\right)^{\frac{1}{2\epsilon}} \quad \text{if } \epsilon < \frac{1}{2} \\ (1 + e^{-1}) \frac{R^2}{\gamma_d^2} \ln \left((1 + e) \frac{R^2}{\gamma_d^2}\right) & \text{if } \epsilon = \frac{1}{2} \\ t_0 (1 - 2(1 - \epsilon) t_0^{1-2\epsilon}), & t_0 \equiv \frac{\epsilon(3-2\epsilon)}{2\epsilon-1} \frac{R^2}{\gamma_d^2} \quad \text{if } \epsilon > \frac{1}{2}. \end{cases}$$

*Proof.* From (2) and (8) we get

$$\|\mathbf{a}_{t+1}\|^2 = \|\mathbf{a}_t\|^2 + 2\mathbf{a}_t \cdot \mathbf{y}_k + \|\mathbf{y}_k\|^2 \leq \|\mathbf{a}_t\|^2 \left(1 + \frac{2(1-\epsilon)}{t}\right) + R^2. \quad (12)$$

Let us assume that  $\epsilon < 1/2$ . Then, using the inequality  $(1+x)^\zeta \geq 1 + \zeta x$  for  $x \geq 0$ ,  $\zeta = 2(1-\epsilon) \geq 1$  in (12) we obtain

$$\|\mathbf{a}_{t+1}\|^2 \leq \|\mathbf{a}_t\|^2 \left(1 + \frac{1}{t}\right)^{2(1-\epsilon)} + R^2$$

from where by dividing both sides with  $(t+1)^{2(1-\epsilon)}$  we arrive at

$$\frac{\|\mathbf{a}_{t+1}\|^2}{(t+1)^{2(1-\epsilon)}} - \frac{\|\mathbf{a}_t\|^2}{t^{2(1-\epsilon)}} \leq \frac{R^2}{(t+1)^{2(1-\epsilon)}}.$$

A repeated application of the above inequality  $t - N$  times ( $t > N \geq 1$ ) gives

$$\begin{aligned} \frac{\|\mathbf{a}_t\|^2}{t^{2(1-\epsilon)}} - \frac{\|\mathbf{a}_N\|^2}{N^{2(1-\epsilon)}} &\leq R^2 \sum_{k=N+1}^t k^{-2(1-\epsilon)} \leq R^2 \int_N^t k^{-2(1-\epsilon)} dk \\ &= \frac{R^2 N^{2\epsilon-1}}{2\epsilon-1} \left( \left(\frac{t}{N}\right)^{2\epsilon-1} - 1 \right). \end{aligned} \quad (13)$$

Now, let us define

$$\alpha_t \equiv \frac{\|\mathbf{a}_t\|}{Rt}$$

and observe that the bounds  $\|\mathbf{a}_t\| \leq Rt$  and  $\|\mathbf{a}_t\| \geq \gamma_d t$  confine  $\alpha_t$  to lie in the range

$$\frac{\gamma_d}{R} \leq \alpha_t \leq 1.$$

Setting  $\|\mathbf{a}_N\| = \alpha_N RN$  in (13) we get the following upper bound on  $\|\mathbf{a}_t\|^2$

$$\|\mathbf{a}_t\|^2 \leq t^{2(1-\epsilon)} \alpha_N^2 R^2 N^{2\epsilon} \left\{ 1 + \frac{\alpha_N^{-2} N^{-1}}{2\epsilon-1} \left( \left(\frac{t}{N}\right)^{2\epsilon-1} - 1 \right) \right\}$$

which combined with the lower bound  $\|\mathbf{a}_t\|^2 \geq \gamma_d^2 t^2$  leads to

$$t^{2\epsilon} \leq \alpha_N^2 \frac{R^2}{\gamma_d^2} N^{2\epsilon} \left\{ 1 + \frac{\alpha_N^{-2} N^{-1}}{2\epsilon - 1} \left( \left( \frac{t}{N} \right)^{2\epsilon-1} - 1 \right) \right\} . \quad (14)$$

For  $\epsilon < 1/2$  the term proportional to  $(t/N)^{2\epsilon-1}$  in (14) is negative and may be dropped to a first approximation leading to the looser upper bound  $t_0$

$$t_0 \equiv N \left( \alpha_N \frac{R}{\gamma_d} \right)^{\frac{1}{\epsilon}} \left( 1 + \frac{\alpha_N^{-2} N^{-1}}{1 - 2\epsilon} \right)^{\frac{1}{2\epsilon}} \quad (15)$$

on the number  $t$  of updates. Then, we may replace  $t$  with its upper bound  $t_0$  in the r.h.s. of (14) and get the improved bound

$$t \leq t_0 \left( 1 - \frac{1}{1 - 2\epsilon} \frac{R^2}{\gamma_d^2} t_0^{-1} \right)^{\frac{1}{2\epsilon}} .$$

This is allowed given that the term proportional to  $(t/N)^{2\epsilon-1}$  in (14) is negative and moreover  $t$  is raised to a negative power. Choosing  $N = \lceil \epsilon^{-1} \rceil$  and  $\alpha_N = 1$  (i.e., setting  $\alpha_N$  to its upper bound which is the least favorable assumption) we obtain the bound stated in Theorem 2 for  $\epsilon < 1/2$ .

Now, let  $\epsilon > 1/2$ . Then, using the inequality  $(1+x)^\zeta + \zeta(1-\zeta)x^2/2 \geq 1 + \zeta x$  for  $x \geq 0$ ,  $0 \leq \zeta = 2(1-\epsilon) \leq 1$  in (12) and the bound  $\|\mathbf{a}_t\| \leq Rt$  we obtain

$$\begin{aligned} \|\mathbf{a}_{t+1}\|^2 &\leq \|\mathbf{a}_t\|^2 \left( 1 + \frac{1}{t} \right)^{2(1-\epsilon)} + (1-\epsilon)(2\epsilon-1) \frac{\|\mathbf{a}_t\|^2}{t^2} + R^2 \\ &\leq \|\mathbf{a}_t\|^2 \left( 1 + \frac{1}{t} \right)^{2(1-\epsilon)} + \epsilon(3-2\epsilon)R^2 . \end{aligned}$$

By dividing both sides of the above inequality with  $(t+1)^{2(1-\epsilon)}$  we arrive at

$$\frac{\|\mathbf{a}_{t+1}\|^2}{(t+1)^{2(1-\epsilon)}} - \frac{\|\mathbf{a}_t\|^2}{t^{2(1-\epsilon)}} \leq \epsilon(3-2\epsilon) \frac{R^2}{(t+1)^{2(1-\epsilon)}} \quad (16)$$

a repeated application of which, using also  $\|\mathbf{a}_1\|^2 \leq R^2 \leq \epsilon(3-2\epsilon)R^2$ , gives

$$\begin{aligned} \frac{\|\mathbf{a}_t\|^2}{t^{2(1-\epsilon)}} &\leq \epsilon(3-2\epsilon)R^2 \sum_{k=1}^t k^{-2(1-\epsilon)} \leq \epsilon(3-2\epsilon)R^2 \left( 1 + \int_1^t k^{-2(1-\epsilon)} dk \right) \\ &= \epsilon(3-2\epsilon)R^2 \left( 1 + \frac{t^{2\epsilon-1} - 1}{2\epsilon - 1} \right) . \end{aligned}$$

Combining the above bound with the bound  $\|\mathbf{a}_t\|^2 \geq \gamma_d^2 t^2$  we obtain

$$t^{2\epsilon} \leq \epsilon(3-2\epsilon) \frac{R^2}{\gamma_d^2} \left( 1 + \frac{t^{2\epsilon-1} - 1}{2\epsilon - 1} \right) \quad (17)$$



or

$$t \leq \frac{\epsilon(3-2\epsilon)}{2\epsilon-1} \frac{R^2}{\gamma_d^2} (1 - 2(1-\epsilon)t^{1-2\epsilon}) \quad . \quad (18)$$

For  $\epsilon > 1/2$  the term proportional to  $t^{1-2\epsilon}$  in (18) is negative and may be dropped to a first approximation leading to the looser upper bound  $t_0$

$$t_0 \equiv \frac{\epsilon(3-2\epsilon)}{2\epsilon-1} \frac{R^2}{\gamma_d^2}$$

on the number  $t$  of updates. Then, we may replace  $t$  with its upper bound  $t_0$  in the r.h.s. of (18) and get the improved bound stated in Theorem 2 for  $\epsilon > 1/2$ . This is allowed given that the term proportional to  $t^{1-2\epsilon}$  in (18) is negative and moreover  $t$  is raised to a negative power.

Finally, taking the limit  $\epsilon \rightarrow 1/2$  in (14) (with  $N = 1$ ,  $\alpha_N = 1$ ) or in (17) we get

$$t \leq \frac{R^2}{\gamma_d^2} (1 + \ln t)$$

which on account of Lemma 1 leads to the bound of Theorem 2 for  $\epsilon = 1/2$ .  $\square$

*Remark 1.* The bound of Theorem 2 holds for PFM as well on account of (4).

The worst-case bound of Theorem 2 for  $\epsilon \ll 1$  behaves like  $\epsilon^{-1}(R/\gamma_d)^{\frac{1}{\epsilon}}$  which suggests an extremely slow convergence if we require margins close to the maximum. From expression (15) for  $t_0$ , however, it becomes apparent that a more favorable assumption concerning the value of  $\alpha_N$  (e.g.,  $\alpha_N \ll 1$  or even as low as  $\alpha_N \sim \gamma_d/R$ ) after the first  $N \gg \alpha_N^{-2}$  updates does lead to tremendous improvement provided, of course, that  $N$  is not extremely large. Such a sharp decrease of  $\|\mathbf{a}_t\|/t$  in the early stages of the algorithm, which may be expected from relation (6) and the discussion that followed, lies behind its experimentally exhibited rather fast convergence.

It would be interesting to find a procedure by which the algorithm will be forced to a guaranteed sharp decrease of the ratio  $\|\mathbf{a}_t\|/t$ . The following two observations will be vital in devising such a procedure. First, we notice that when PDM with accuracy parameter  $\epsilon$  has converged in  $t_c$  updates the threshold  $(1-\epsilon)\|\mathbf{a}_{t_c}\|^2/t_c$  of the misclassification condition must have fallen below  $\gamma_d \|\mathbf{a}_{t_c}\|$ . Otherwise, the normalized margin  $\mathbf{u}_{t_c} \cdot \mathbf{y}_k$  of all patterns  $\mathbf{y}_k$  would be larger than  $\gamma_d$ . Thus,  $\alpha_{t_c} < (1-\epsilon)^{-1}\gamma_d/R$ . Second, after convergence of the algorithm with accuracy parameter  $\epsilon_1$  in  $t_{c_1}$  updates we may lower the accuracy parameter from the value  $\epsilon_1$  to the value  $\epsilon_2$  and continue the run from the point where convergence with parameter  $\epsilon_1$  has taken place since for all updates that took place during the first run the misclassified patterns would certainly satisfy (at that time) the condition associated with the smaller parameter  $\epsilon_2$ . This way, the first run is legitimately fully incorporated into the second one and the  $t_{c_1}$  updates required for convergence during the first run may be considered the first  $t_{c_1}$  updates of the second run under this specific policy of presenting patterns to the algorithm. Combining the above two observations we see that by employing

a first run with accuracy parameter  $\epsilon_1$  we force the algorithm with accuracy parameter  $\epsilon_2 < \epsilon_1$  to have  $\alpha_t$  decreased from a value  $\sim 1$  to a value  $\alpha_{t_{c_1}} < (1 - \epsilon_1)^{-1} \gamma_d / R$  in the first  $t_{c_1}$  updates.

The above discussion suggests that we consider a decreasing sequence of parameters  $\epsilon_n$  such that  $\epsilon_{n+1} = \epsilon_n / \eta$  ( $\eta > 1$ ) starting with  $\epsilon_0 = 1/2$  and ending with the required accuracy  $\epsilon$  and perform successive runs of PDM with accuracies  $\epsilon_n$  until convergence in  $t_{c_n}$  updates is reached. According to our earlier discussion  $t_{c_n}$  includes the updates that led the algorithm to convergence in the current and all previous runs. Moreover, at the end of the run with parameter  $\epsilon_n$  we will have ensured that  $\alpha_{t_{c_n}} < (1 - \epsilon_n)^{-1} \gamma_d / R$ . Therefore,  $t_{c_{n+1}}$  satisfies  $t_{c_{n+1}} \leq t_0$  or

$$t_{c_{n+1}} \leq t_{c_n} \left( \frac{1}{1 - \epsilon_n} \right)^{\eta/\epsilon_n} \left( 1 + \frac{(1 - \epsilon_n)^2}{1 - 2\epsilon_n/\eta} \frac{R^2}{\gamma_d^2} t_{c_n}^{-1} \right)^{\eta/2\epsilon_n}.$$

This is obtained by substituting in (15) the values  $\epsilon = \epsilon_{n+1} = \epsilon_n / \eta$ ,  $N = t_{c_n}$  and  $\alpha_N = (1 - \epsilon_n)^{-1} \gamma_d / R$  which is the least favorable choice for  $\alpha_{t_{c_n}}$ . Let us assume that  $\epsilon_n \ll 1$  and set  $t_{c_n} = \xi_n^{-1} R^2 / \gamma_d^2$  with  $\xi_n \ll 1$ . Then,  $1/(1 - \epsilon_n)^{\eta/\epsilon_n} \simeq e^\eta$  and

$$\left( 1 + \frac{(1 - \epsilon_n)^2}{1 - 2\epsilon_n/\eta} \frac{R^2}{\gamma_d^2} t_{c_n}^{-1} \right)^{\eta/2\epsilon_n} \simeq (1 + \xi_n)^{\eta/2\epsilon_n} \simeq e^{\eta\xi_n/2\epsilon_n}.$$

For  $\xi_n \simeq \epsilon_n$  the term above becomes approximately  $e^{\eta/2}$  while for  $\xi_n \ll \epsilon_n$  approaches 1. We see that under the assumption that PDM with accuracy parameter  $\epsilon_n$  converges in a number of updates  $\gg R^2 / \gamma_d^2$  the ratio  $t_{c_{n+1}} / t_{c_n}$  in the successive run scenario is rather tightly constrained. If, instead, our assumption is not satisfied then convergence of the algorithm is fast anyway. Notice, that the value of  $t_{c_{n+1}} / t_{c_n}$  inferred from the bound of Theorem 2 is  $\sim \eta (R / \gamma_d)^{(\eta-1)/\epsilon_n}$  which is extremely large. We conclude that PDM employing the successive run scenario (PDM-succ) potentially converges in a much smaller number of steps.

## 4 Efficient Implementation

To reduce the computational cost involved in running PDM, we extend the procedure of [14, 13] and construct a three-member nested sequence of reduced “active sets” of data points. As we cycle once through the full dataset, the (largest) first-level active set is formed from the points of the full dataset satisfying  $\mathbf{a}_t \cdot \mathbf{y}_k \leq c_1(1 - \epsilon) \|\mathbf{a}_t\|^2 / t$  with  $c_1 = 2.2$ . Analogously, the second-level active set is formed as we cycle once through the first-level active set from the points which satisfy  $\mathbf{a}_t \cdot \mathbf{y}_k \leq c_2(1 - \epsilon) \|\mathbf{a}_t\|^2 / t$  with  $c_2 = 1.1$ . The third-level active set comprises the points that satisfy  $\mathbf{a}_t \cdot \mathbf{y}_k \leq (1 - \epsilon) \|\mathbf{a}_t\|^2 / t$  as we cycle once through the second-level active set. The third-level active set is presented repetitively to the algorithm for  $N_{\text{ep}_3}$  mini-epochs. Then, the second-level active set is presented  $N_{\text{ep}_2}$  times. During each round involving the second-level set, a new third-level set is constructed and a new cycle of  $N_{\text{ep}_3}$  passes begins. When the number of  $N_{\text{ep}_2}$  cycles involving the second-level set is reached the first-level

set becomes active again leading to the population of a new second-level active set. By invoking the first-level set for the  $(N_{\text{ep1}} + 1)^{\text{th}}$  time, we trigger the loading of the full dataset and the procedure starts all over again until no point is found misclassified among the ones comprising the full dataset. Of course, the  $N_{\text{ep1}}$ ,  $N_{\text{ep2}}$  and  $N_{\text{ep3}}$  rounds are not exhausted if no update takes place during a round. In all experiments we choose  $N_{\text{ep1}} = 9$ ,  $N_{\text{ep2}} = N_{\text{ep3}} = 12$ . In addition, every time we make use of the full dataset we actually employ a permuted instance of it. Evidently, the whole procedure amounts to a different way of sequentially presenting the patterns to the algorithm and does not affect the applicability of our theoretical analysis. A completely analogous procedure is followed for PFM.

An additional mechanism providing a substantial improvement of the computational efficiency is the one of performing multiple updates [14, 13] once a data point is presented to the algorithm. It is understood, of course, that in order for a multiple update to be compatible with our theoretical analysis it should be equivalent to a certain number of updates occurring as a result of repeatedly presenting to the algorithm the data point in question. For PDM when a pattern  $\mathbf{y}_k$  is found to satisfy the misclassification condition (8) we perform  $\lambda = \lceil \mu_+ \rceil + 1$  updates at once. Here,  $\mu_+$  is the smallest non-negative root of the quadratic equation in the variable  $\mu$  derivable from the relation  $(t + \mu)\mathbf{a}_{t+\mu} \cdot \mathbf{y}_k - (1 - \epsilon) \|\mathbf{a}_{t+\mu}\|^2 = 0$  in which  $\mathbf{a}_{t+\mu} \cdot \mathbf{y}_k = \mathbf{a}_t \cdot \mathbf{y}_k + \mu \|\mathbf{y}_k\|^2$  and  $\|\mathbf{a}_{t+\mu}\|^2 = \|\mathbf{a}_t\|^2 + 2\mu\mathbf{a}_t \cdot \mathbf{y}_k + \mu^2 \|\mathbf{y}_k\|^2$ . Thus, we require that as a result of the multiple update the pattern violates the misclassification condition. Similarly, we perform multiple updates for PFM.

Finally, in the case of PDM (no successive runs) when we perform multiple updates we start doing so after the first full epoch. This way, we avoid the excessive growth of the length of the weight vector due to the contribution to the solution of many aligned patterns in the early stages of the algorithm which hinders the fast decrease of  $\|\mathbf{a}_t\|/t$ . Moreover, in this scenario when we select the first-level active set as we go through the full dataset for the first time (first full epoch) we found it useful to set  $c_1 = c_2 = 1.1$  instead of  $c_1 = 2.2$ .

## 5 Experimental Evaluation

We compare PDM with several other large margin classifiers on the basis of their ability to achieve fast convergence to a certain approximation of the “optimal” hyperplane in the feature space where the patterns are linearly separable. For linearly separable data the feature space is the initial instance space whereas for inseparable data (which is the case here) a space extended by as many dimensions as the instances is considered where each instance is placed at a distance  $\Delta$  from the origin in the corresponding dimension<sup>2</sup> [4]. This extension generates a margin of at least  $\Delta/\sqrt{m}$ . Moreover, its employment relies on the well-known

<sup>2</sup>  $\mathbf{y}_k = [\bar{\mathbf{y}}_k, l_k \Delta \delta_{1k}, \dots, l_k \Delta \delta_{mk}]$ , where  $\delta_{ij}$  is Kronecker’s  $\delta$  and  $\bar{\mathbf{y}}_k$  the projection of the  $k^{\text{th}}$  extended instance  $\mathbf{y}_k$  (multiplied by its label  $l_k$ ) onto the initial instance space. The feature space mapping defined by the extension commutes with a possible augmentation (with parameter  $\rho$ ) in which case  $\bar{\mathbf{y}}_k = [l_k \bar{\mathbf{x}}_k, l_k \rho]$ . Here  $\bar{\mathbf{x}}_k$  represents the  $k^{\text{th}}$  data point.

equivalence between the hard margin optimization in the extended space and the soft margin optimization in the initial instance space with objective function  $\|\mathbf{w}\|^2 + \Delta^{-2} \sum_i \bar{\xi}_i^2$  involving the weight vector  $\mathbf{w}$  and the 2-norm of the slacks  $\bar{\xi}_i$  [2]. Of course, all algorithms are required to solve identical hard margin problems.

The datasets we used for training are: the Adult ( $m = 32561$  instances,  $n = 123$  attributes) and Web ( $m = 49749$ ,  $n = 300$ ) UCI datasets as compiled by Platt [15], the training set of the KDD04 Physics dataset ( $m = 50000$ ,  $n = 70$  after removing the 8 columns containing missing features) obtainable from <http://kodiak.cs.cornell.edu/kddcup/datasets.html>, the Real-sim ( $m = 72309$ ,  $n = 20958$ ), News20 ( $m = 19996$ ,  $n = 1355191$ ) and Webspam (unigram treatment with  $m = 350000$ ,  $n = 254$ ) datasets all available at <http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets>, the multiclass Covertype UCI dataset ( $m = 581012$ ,  $n = 54$ ) and the full Reuters RCV1 dataset ( $m = 804414$ ,  $n = 47236$ ) obtainable from [http://www.jmlr.org/papers/volume5/lewis04a/lyrl2004\\_rcv1v2\\_README.htm](http://www.jmlr.org/papers/volume5/lewis04a/lyrl2004_rcv1v2_README.htm). For the Covertype dataset we study the binary classification problem of the first class versus rest while for the RCV1 we consider both the binary text classification tasks of the C11 and CCAT classes versus rest. The Physics and Covertype datasets were rescaled by a multiplicative factor 0.001. The experiments were conducted on a 2.5 GHz Intel Core 2 Duo processor with 3 GB RAM running Windows Vista. Our codes written in C++ were compiled using the g++ compiler under Cygwin.

The parameter  $\Delta$  of the extended space is chosen from the set  $\{3, 1, 0.3, 0.1\}$  in such a way that it corresponds approximately to  $R/10$  or  $R/3$  depending on the size of the dataset such that the ratio  $\gamma_d/R$  does not become too small (given that the extension generates a margin of at least  $\Delta/\sqrt{m}$ ). More specifically, we have chosen  $\Delta = 3$  for Covertype,  $\Delta = 1$  for Adult, Web and Physics,  $\Delta = 0.3$  for Webspam, C11 and CCAT and  $\Delta = 0.1$  for Real-sim and News20. We also verified that smaller values of  $\Delta$  do not lead to a significant decrease of the training error. For all datasets and for algorithms that introduce bias through augmentation the associated parameter  $\rho$  was set to the value  $\rho = 1$ .

We begin our experimental evaluation by comparing PDM with PFM. We run PDM with accuracy parameter  $\epsilon = 0.01$  and subsequently PFM with the fixed margin  $\beta = (1 - \epsilon)\gamma_d$  set to the value  $\gamma'_d$  of the directional margin achieved by PDM. This procedure is repeated using PDM-succ with step  $\eta = 8$  (i.e.,  $\epsilon_0 = 0.5$ ,  $\epsilon_1 = 0.0625$ ,  $\epsilon_2 = \epsilon = 0.01$ ). Our results (the value of the directional margin  $\gamma'_d$  achieved, the number of required updates (upd) for convergence and the CPU time for training in seconds (s)) are presented in Table 1. We see that PDM is considerably faster than PFM as far as training time is concerned in spite of the fact that PFM needs much less updates for convergence. The successive run scenario succeeds, in accordance with our expectations, in reducing the number of updates to the level of the updates needed by PFM in order to achieve the same value of  $\gamma'_d$  at the expense of an increased runtime. We believe that it is fair to say that PDM-succ with  $\eta = 8$  has the overall performance of PFM without the defect of the need for a priori knowledge of the value of  $\gamma_d$ . We also notice that although the accuracy  $\epsilon$  is set to the same value for both scenarios

**Table 1.** Results of an experimental evaluation comparing the algorithms PDM and PDM-succ with PFM.

data set	PDM $\epsilon = 0.01$			PFM		PDM-succ $\epsilon = 0.01$			PFM	
	$10^4\gamma'_d$	$10^{-6}\text{upd}$	s	$10^{-6}\text{upd}$	s	$10^4\gamma'_d$	$10^{-6}\text{upd}$	s	$10^{-6}\text{upd}$	s
Adult	84.57	27.43	3.7	10.70	7.3	84.46	9.312	5.3	9.367	6.6
Web	209.6	739.4	0.8	1.089	0.9	209.1	0.838	0.9	0.871	0.8
Physics	44.54	9.449	10.4	6.021	13.8	44.53	5.984	15.3	6.006	13.8
Real-sim	39.93	15.42	13.6	12.69	35.7	39.74	5.314	13.8	5.306	14.3
News20	91.90	2.403	27.4	1.060	55.6	91.68	0.814	47.7	0.813	43.7
Webspam	10.05	331.0	197.5	108.4	348.0	10.03	89.72	247.0	89.60	264.5
Coverttype	47.51	189.7	86.6	68.86	156.0	47.48	66.03	146.1	64.41	142.5
C11	13.81	148.6	156.3	75.26	895.1	13.77	49.02	612.4	49.22	557.5
CCAT	9.279	307.7	310.6	151.2	1923.5	9.253	107.8	1389.8	107.8	1601.0

the margin achieved with successive runs is lower. This is an indication that PDM-succ obtains a better estimate of the maximum directional margin  $\gamma_d$ .

We also considered other large margin classifiers representing classes of algorithms such as perceptron-like algorithms, decomposition SVMs and linear SVMs with the additional requirement that the chosen algorithms need only specification of an accuracy parameter. From the class of perceptron-like algorithms we have chosen (aggressive) ROMMA which is much faster than ALMA in the light of the results presented in [9, 14]. Decomposition SVMs are represented by SVM<sup>light</sup> [7] which, apart from being one of the fastest algorithms of this class, has the additional advantage of making very efficient use of memory, thereby making possible the training on very large datasets. Finally, from the more recent class of linear SVMs we have included in our study the dual coordinate descent (DCD) algorithm [8] and the margin perceptron with unlearning (MPU)<sup>3</sup> [13]. We considered the DCD versions with 1-norm (DCD-L1) and 2-norm (DCD-L2) soft margin which for the same value of the accuracy parameter produce identical solutions if the penalty parameter is  $C = \infty$  for DCD-L1 and  $C = 1/(2\Delta^2)$  for DCD-L2. The source for SVM<sup>light</sup> (version 6.02) is available at <http://smvlight.joachims.org> and for DCD at <http://www.csie.ntu.edu.tw/~cjlin/liblinear>. The absence of publicly available implementations for ROMMA necessitated the writing of our own code in C++ employing the mechanism of active sets proposed in [9] and incorporating a mechanism of permutations performed at the beginning of a full epoch. For MPU the implementation followed closely [13] with active set parameters  $\bar{c} = 1.01$ ,  $N_{\text{ep}_1} = N_{\text{ep}_2} = 5$ , gap parameter  $\delta b = 3R^2$  and early stopping.

The experimental results (margin values achieved and training runtimes) involving the above algorithms with the accuracy parameter set to 0.01 for all of

<sup>3</sup> MPU uses dual variables but is not formulated as an optimization. It is a perceptron incorporating a mechanism of reduction of possible contributions from “very-well classified” patterns to the weight vector which is an essential ingredient of SVMs.

**Table 2.** Results of experiments with ROMMA, SVM<sup>light</sup>, DCD-L1, DCD-L2 and MPU algorithms. The accuracy parameter for all algorithms is set to 0.01.

data set	ROMMA		SVM <sup>light</sup>		DCD-L1		DCD-L2	MPU	
	$10^4\gamma'_d$	s	$10^4\gamma'$	s	$10^4\gamma'_d$	s	s	$10^4\gamma'_d$	s
Adult	84.66	275.8	84.90	414.2	84.95	0.6	0.5	84.61	0.8
Web	209.6	52.6	209.4	40.3	209.5	0.7	0.6	209.5	0.3
Physics	44.57	117.7	44.60	2341.8	44.57	22.5	20.0	44.62	4.9
Real-sim	39.89	1318.8	39.80	146.5	39.81	6.4	5.6	39.78	3.3
News20	92.01	4754.0	91.95	113.8	92.17	48.1	47.1	91.62	15.8
Webspam	10.06	39760.6	10.07	29219.4	10.08	37.5	33.0	10.06	28.2
Coverttype	47.54	43282.0	47.73	48460.3	47.71	18.1	15.0	47.67	18.7
C11	13.82	146529.2	13.82	20127.8	13.83	30.7	27.2	13.79	20.2
CCAT	9.290	298159.4	9.291	83302.4	9.303	51.9	46.2	9.264	36.1

them are summarized in Table 2. Notice that for SVM<sup>light</sup> we give the geometric margin  $\gamma'$  instead of the directional one  $\gamma'_d$  because SVM<sup>light</sup> does not introduce bias through augmentation. For the rest of the algorithms considered, including PDM and PFM, the geometric margin  $\gamma'$  achieved is not listed in the tables since it is very close to the directional margin  $\gamma'_d$  if the augmentation parameter  $\rho$  is set to the value  $\rho = 1$ . Moreover, for DCD-L1 and DCD-L2 the margin values coincide as we pointed out earlier. From Table 2 it is apparent that ROMMA and SVM<sup>light</sup> are orders of magnitude slower than DCD and MPU. Comparing the results of Table 1 with those of Table 2 we see that PDM is orders of magnitude faster than ROMMA which is its natural competitor since they both belong to the class of perceptron-like algorithms. PDM is also much faster than SVM<sup>light</sup> but statistically a few times slower than DCD, especially for the larger datasets. Moreover, PDM is a few times slower than MPU for all datasets. Finally, we observe that the accuracy achieved by PDM is, in general, closer to the before-run accuracy 0.01 since in most cases PDM obtains lower margin values. This indicates that PDM succeeds in obtaining a better estimate of the maximum margin than the remaining algorithms with the possible exception of MPU.

Before we conclude our comparative study it is fair to point out that PDM is not the fastest perceptron-like large margin classifier. From the results of [14] the fastest algorithm of this class is the margitron which has strong before-run guarantees and a very good after-run estimate of the achieved accuracy through (5). However, its drawback is that an approximate knowledge of the value of  $\gamma_d$  (preferably an upper bound) is required in order to fix the parameter controlling the margin threshold. Although there is a procedure to obtain this information, taking all the facts into account the employment of PDM seems preferable.

## 6 Conclusions

We introduced the perceptron with dynamic margin (PDM), a new approximate maximum margin classifier employing the classical perceptron update, demonstrated its convergence in a finite number of steps and derived an upper bound on them. PDM uses the required accuracy as the only input parameter. Moreover, it is a strictly online algorithm in the sense that it decides whether to perform an update taking into account only its current state and irrespective of whether the pattern presented to it has been encountered before in the process of cycling repeatedly through the dataset. This certainly does not hold for linear SVMs. Our experimental results indicate that PDM is the fastest large margin classifier enjoying the above two very desirable properties.

## References

1. Blum, A.: Lectures on machine learning theory. Carnegie Mellon University, USA. Available at <http://www.cs.cmu.edu/~avrim/ML09/lect0126.pdf>
2. Cristianini, N., Shawe-Taylor, J.: An introduction to support vector machines (2000) Cambridge, UK: Cambridge University Press
3. Duda, R.O., Hart, P.E.: Pattern classification and scene analysis (1973) Wiley
4. Freund, Y., Shapire, R.E.: Large margin classification using the perceptron algorithm. *Machine Learning* **37(3)** (1999) 277–296
5. Gentile, C.: A new approximate maximal margin classification algorithm. *Journal of Machine Learning Research* **2** (2001) 213–242
6. Joachims, T.: Making large-scale SVM learning practical. In *Advances in kernel methods-support vector learning* (1999) MIT Press
7. Joachims, T.: Training linear SVMs in linear time. *KDD* (2006) 217–226
8. Hsieh, C.-J., Chang, K.-W., Lin, C.-J., Keerthi, S.S., Sundararajan, S.: A dual coordinate descent method for large-scale linear SVM. *ICML* (2008) 408–415
9. Ishibashi, K., Hatano, K., Takeda, M.: Online learning of maximum p-norm margin classifiers. *COLT* (2008) 69–80.
10. Krauth, W., Mézard, M.: Learning algorithms with optimal stability in neural networks. *Journal of Physics* **A20** (1987) L745–L752
11. Li, Y., Long, P.: The relaxed online maximum margin algorithm. *Machine Learning*, **46(1-3)** (2002) 361–387
12. Novikoff, A.B.J.: On convergence proofs on perceptrons. In *Proc. Symp. Math. Theory Automata*, Vol. 12 (1962) 615–622
13. Panagiotakopoulos, C., Tsampouka, P.: The margin perceptron with unlearning. *ICML* (2010) 855–862
14. Panagiotakopoulos, C., Tsampouka, P.: The margitron: A generalized perceptron with margin. *IEEE Transactions on Neural Networks* **22(3)** (2011) 395–407
15. Platt, J.C.: Sequential minimal optimization: A fast algorithm for training support vector machines. Microsoft Res. Redmond WA, Tech. Rep. MSR-TR-98-14 (1998)
16. Rosenblatt, F.: The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, **65(6)** (1958) 386–408
17. Tsampouka, P., Shawe-Taylor, J.: Perceptron-like large margin classifiers. Tech. Rep., ECS, University of Southampton, UK (2005). Obtainable from <http://eprints.ecs.soton.ac.uk/10657>

18. Tsampouka, P., Shawe-Taylor, J.: Analysis of generic perceptron-like large margin classifiers. ECML (2005) 750–758
19. Tsampouka, P., Shawe-Taylor, J.: Constant rate approximate maximum margin algorithms. ECML (2006) 437–448
20. Tsampouka, P., Shawe-Taylor, J.: Approximate maximum margin algorithms with rules controlled by the number of mistakes. ICML (2007) 903–910
21. Vapnik, V.: Statistical learning theory (1998) Wiley